Hunt or Be Hunted

Introducere

- Nume: Glodariu Ana
- Grupa: 331CB
- Îndrumător: Ionuț Oțelea
- Proiectul constă într-un joc de supraviețuire într-un tărâm terorizat de un monstru.
- Ideea stă în crearea unei experiențe de dezvoltare pline de distracție pentru mine, dezvoltatorul, în timp ce ofer utilizatorilor o experiență de joc captivantă și plină de bucurie odată ce proiectul este finalizat.

Proiectul este inspirat din jocul **Hunt the Wumpus**, iar eu intenționez să îi aduc un amestec captivant de *basm* și *horror*. Astfel, m-am decis să plasez jucătorul într-un tărâm al florilor îndepărtat, în care trăiau Împăratul Bujorel și singura lui fiică Trandafirica.

Cu toate că acest tărâm fermecător este înconjurat de frumusețea florilor, el ascunde și un pericol teribil: un monstru care devorează totul în calea sa. Monstrul a distrus toate florile, lăsând în urmă doar pământ arid și dezolat. Împăratul, simțindu-se neputincios, a anunțat în întreaga țară că orice erou care va reuși să învingă bestia va primi mâna fiicei sale în căsătorie. Cu toate acestea, se spunea că Trandafirica era de o urâțenie rară, iar niciun voinic nu îndrăznise să se confrunte cu monstrul.

Oare care va fi primul jucător care își va pune viața în pericol indiferent de premiu? Cine știe, poate zvonurile sunt false și adevărata frumusețe a Trandafiricii se ascunde dincolo de povestea spusă de gurile rele.

Descriere generală

Schemă bloc:

Funcționalități module proiect:

- ecranul oled e folosit pentru grafica jocului & timer
- matricea de led-uri reprezintă harta jocului
- buzzer-ul contribuie la localizarea monstrului & cufăr de comori
- joystick-ul e pentru deplasarea prin harta sub formă de matrice

Matricea va fi de dimensiune 8×8, fiecare led reprezentând o poziție validă a jucătorului, una dintre poziții va fi ocupată de monstru și alta de cufărul cu arme necesar pentru a ucide bestia și a câștiga jocul.

În funcție de inputul dat de jucător joystick-ului, acesta se poate mișca pe orizontală și verticală pe hartă, iar folosind funcționalitatea de switch al joystick-ului el poate da hit monstrului. Pentru a da hit monstrului, jucătorul trebuie mai întâi să-l localizeze, să se afle pe o poziție de pe hartă alăturată lui pe axele ox sau oy îndreptat către el și să apese pe joystick.

Buzzer-ul va fi activat doar atunci când jucătorul este în apropierea monstrului pentru a contribui la localizarea acestuia. Acesta poate fi folosit și la redarea unor sunete când jucătorul pierde sau câștigă.

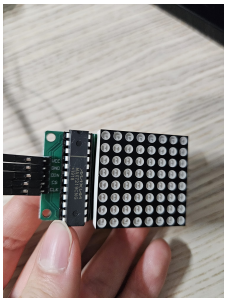
Ecranul OLED contribuie dpdv al graficii jocului, afișarea de mesaje și a unui timer. Scopul timer-ului este de a da un sentiment de survival și presiune, determinând jucătorul să învingă bestia într-un timp limitat.

Pentru a face proiectul portabil, folosesc o baterie de 9V pentru alimentarea microcontroller-ului ATmega328P, în loc să îl conectez la laptop prin USB. Placa va comunica cu ecranul oled prin I2C și cu matricea de led-uri prin SPI în funcție de inputul primit de la joystick. La rândul lui, joystick-ul va fi conectat la 2 pini analogici pentru deplasarea pe verticală și orizontală, și la 1 pin digital pentru un comportament asemănător unui buton. Buzzer-ul este conectat la un pin digital împreună cu un rezistor de 100Ω.

Hardware Design

- Schemă de componente:

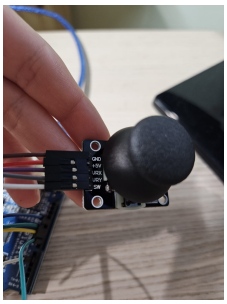
- Listă de piese:
  - Arduino Uno (ATmega328P)
  - Modul MAX7219 LED Dot Matrix
  - Modul buzzer pasiv
  - Male-Female, Female-Female, Male-Male wires
  - 9V Battery Connector with DC Jack
  - baterie de 9V
  - Joystick Breakout Board
  - Modul OLED (128×32 px)
  - Rezistori



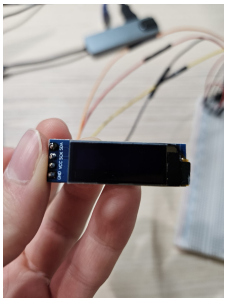
- Conectare matrice de leduri - Arduino Uno (comunicare prin SPI):
  - VCC - 5V
  - GND - GND
  - DIN - PIN11
  - CS - PIN10
  - CLK - PIN13

PIN13 de pe Arduino Uno este un pin SCK (Serial Data Clock) folosit pentru comunicarea prin SPI.

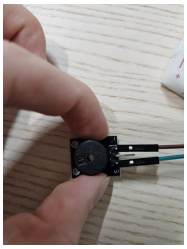
Pinul de DOUT de pe modulul matrice de leduri este pentru conectarea în lanț a mai multor matrice și nu-l voi folosi pentru proiect.



- Conectare joystick - Arduino Uno
  - GND - GND
  - +5V - 5V
  - VRX - A0 (pin analogic)
  - VRY - A1 (pin analogic)
  - SW - PIN2 (pin digital)

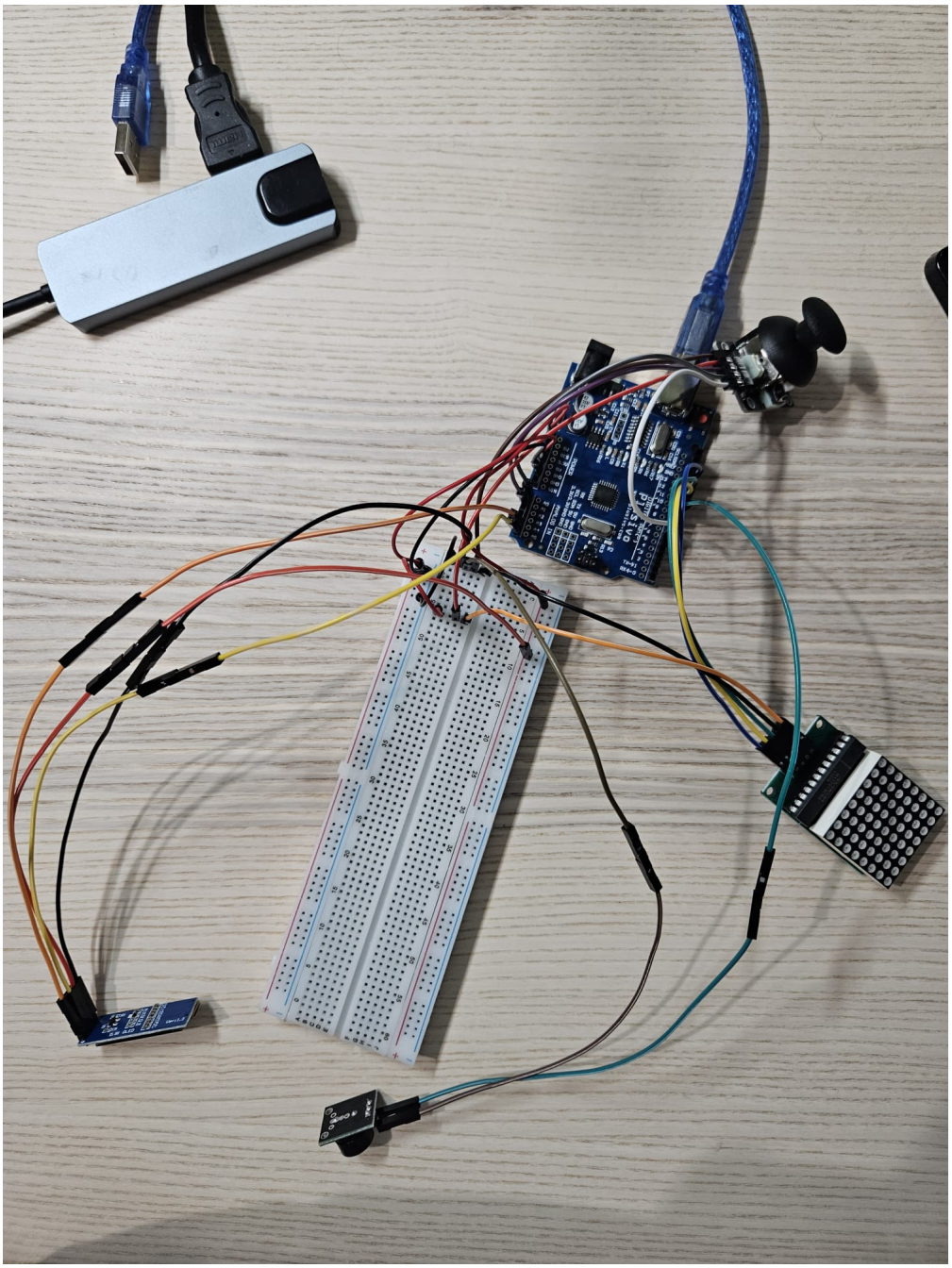


- Conectare ecran OLED - Arduino Uno (comunicare prin I2C)
  - GND - GND
  - VCC - 5V
  - SDA (serial data pin)- A4 (folosit ca SDA pentru comunicarea prin I2C)
  - SCL (clock pin) - A5 (folosit ca SCL pentru comunicarea prin I2C)

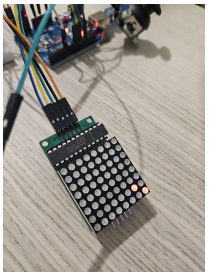


- Conectare buzzer pasiv - Arduino Uno
  - GND - GND
  - Signal Input Pin - PIN9

Al treilea pin nu se conectează la nimic.



- Testare funcționalitate matrice → verific că pot aprinde leduri independent unele de alte în matricea de leduri și citesc și valorile din pinii analogici legați la joystick



Software Design

- mediu de dezvoltare: Arduino IDE
- librării şi surse 3rd-party:
  - SPI.h
  - Wire.h
  - Adafruit_GFX.h → pentru ecranul oled
  - Adafruit_SSD1306.h → pentru ecranul oled
  - MD_MAX72xx.h →pentru matricea de leduri
- algoritmi şi structuri pe care plănuiesc să le implementez:
  - inițializarea matricei de joc cu monstrul, chest-ul
  - update-ul lumii cu poziția curentă a jucătorului și drumul parcurs până atunci
  - când jucătorul se află în jurul monstrului sau a chest-ului, acesta va fi alertat de buzzer și de apariția pe ecranul oled a unei imagini pixelate corespunzătoare
  - timer-ul implementat crește dificultatea jocului - jucătorul având un timp limitat pentru a omorî monstrul
  - afișarea timer-ului pe ecran

În funcția de setup setez pinii, inițializez matricea de led-uri, ecranul OLED și creez harta - adică plasez monstrul și chest-ul într-o poziție random. Jucătorul va fi plasat la începutul fiecărui joc în colțul stânga jos.

Pentru funcția de switch al joystick-ului am creeat o **întrerupere**, care va avea loc de fiecare dată când apăs pe joystick.

Pentru **timer** configurez Timer0 al plăcuței pentru a funcționa în modul CTC (Clear Timer on Compare Match), setează un prescaler de 64 și configurează timerul pentru a genera o întrerupere la fiecare 1 milisecundă.

Pentru ca jucătorul să se deplaseze am definit funcția **getDirection** care primește inputul de la pinii VRX și VRY ai joystick-ului, și în funcție de valorile analogice primite întoarce una dintre direcțiile sus, jos, stânga, dreapta.

Funcția **dangerZone()** alertează jucătorul când se află în apropierea monstrului cu ajutorul buzzer-ului și al ecranului OLED.

Funcția **chestZone()** alertează jucătorul când se află în apropierea cufărului cu ajutorul buzzer-ului și al ecranului OLED.

Funcția **checkHazard()** verifică dacă jucătorul a picat fix pe poziția monstrului.

Funcția **checkLoot()** verifică dacă jucătorul a colectat arma din cufăr.

Am definite niște variabile de tip bool și în funcție de valoarea lor, pe ecran va apărea o imagine cu eroul, prințesa (în cazul în care câștigă), cufărul (când se află în preajma lui), sabia (când deschide cufărul), monstrul (când se află în preajma lui) sau un mormânt (când eroul este mâncat de monstru). Imaginile pixelate au fost create în GIMP și după convertite în bitmap cu ajutorul site-ului: https://javl.github.io/image2cpp/

Eroul **pierde** jocul în următoarele situații:

- a rămas fără timp (au trecut 60 de secunde)

- a încercat să lovească monstrul și nu l-a nimerit (monstrul are un auz foarte dezvoltat și va veni să atace eroul, astfel va fi omorât)
- a încercat să atace monstrul fără o armă
- dă de monstru fără să-l atace

Eroul **câștigă** jocul dacă localizează mai întâi arma, iar odată localizat și monstrul va trebui să lovească în sensul și direcția lui. Pentru a ne da seama în ce sens lovește eroul, în cod reținem în variabila lastMove sensul din care vine jucătorul înspre monstru (de sus, de jos, din stânga sau din dreapta). Eroul trebuie să omoare monstrul în timpul alocat pentru a câștiga.

De fiecare dată când jucătorul pierde sau câștigă, **jocul se resetează**.

- void(* reset) (void) = 0; = declanșează execuția codului de la adresa 0, ce este echivalentă cu o resetare a sistemului.

Pentru afișarea drumului parcurs de jucător folosesc o matrice de 8×8, pozițiile vizitate luând valoarea 1, iar pentru ca jucătorul să știe poziția curentă, led-ul pe care se află va clipi.

De asemenea, ca atunci când jucătorul se mișcă stânga, dreapta, sus sau jos, acesta să se deplaseze cu o singură poziție în sensul corespunzător, am simulat un delay de o secundă între 2 deplasări, folosindu-mă de contorul utilizat la timer (contorul este incrementat de fiecare dată când întreruperea este generată).

Rezultate Obţinute

Rezultat final:

https://drive.google.com/file/d/1-zmoEcbHJrUvFPrlUK8Y_qhrAqo7qiL4/view?usp=sharing

Concluzii

Proiectul ar putea fi puțin mai complex din punct de vedere al numărului de monștri sau prin adăugarea de animații pe ecranul OLED (asta ținând cont și de spațiul de stocare al programului (flash-ul) și de memoria dinamică folosită) - programul meu curent deja ocupând destul de mult:

- '*Sketch uses 23548 bytes (72%) of program storage space. Maximum is 32384 bytes.*'
- '*Global variables use 856 bytes (41%) of dynamic memory, leaving 1192 bytes for local variables. Maximum is 2048 bytes.*'

De asemenea, am întâmpinat probleme din cauza folosirii timer-ului 0, pentru că nu am mai putut folosi funcția de delay() și nici millis(), de care aș fi avut nevoie la creearea sunetelor cu ajutorul buzzer-ului, la care am găsit o alternativă nu atât de bună (zic eu) - generam o anumită frecvență pe un interval determinat de valoarea contorului.

Cu toate acestea, chiar dacă am întâmpinat unele probleme, a fost plăcut să gasesc soluții la ele de una singură. huntorbehunted.zip

Download

huntorbehunted.zip (arhiva încă conține cod pentru afișări de date seriale în cazul în care mai e nevoie de debugging)

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

- 04.05.2024 - creare pagină de prezentare a proiectului, adăugare descriere, schemă bloc, listă de piese, schemă în Wokwi
- 16.05.2024 - implementare hardware, conectarea pinilor, verificarea funcționării a unei componente
- 23.05.2024 - finisare implementare software
- 25.05.2014 - încărcare video proiect + arhivă

Bibliografie/Resurse

Inspirație:

- https://ro.wikipedia.org/wiki/Hunt_the_Wumpus

Resurse hardware:

- https://docs.wokwi.com/parts/board-ssd1306
- https://arduinomodules.info/ky-006-passive-buzzer-module
- https://docs.wokwi.com/parts/wokwi-analog-joystick
- https://docs.wokwi.com/parts/wokwi-max7219-matrix
- https://majicdesigns.github.io/MD_MAX72XX/_m_d___m_a_x72xx_8h.html

Resurse software:

- https://javl.github.io/image2cpp/

→ pentru a converti imagini în byte arrays

- https://ocw.cs.pub.ro/courses/pm/lab/lab2-2023/

→ pentru implementare întreruperi

- https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023-2024/

→ pentru implementare timer

- https://ocw.cs.pub.ro/courses/pm/lab/lab5-2023-2024/

→ pentru comunicare SPI

- https://ocw.cs.pub.ro/courses/pm/lab/lab6-2023-2024/

→ pentru comunicare I2C

Export to PDF